DATA PROCESSING SYSTEM AND METHOD

Field of the Invention

5   The present invention relates to a data processing
system and method and, more particularly, to such a
system and method for solving polynomial equations such
as, for example, those used in implementing a Berlekamp-
Massey algorithm.

10

Background to the Invention

       Modern communication systems use a variety of data
processing techniques to ensure that data received at a
15   receiver can be correctly decoded, that is, comprehended.
Various forms of error detection and error correction
codes have been developed that add redundancy to message
symbols that can be used to detect and correct errors in
the code word up to a given limit or merely to detect
20   errors up to a larger limit. Bose-Chaudhuri-Hocquenghem
(BCH) codes and Reed-Solomon (RS) codes are among the
most widely used block codes for communication systems
and storage systems. The mathematics underlying BCH and
RS codes is explained in, for example, E.R. Berlekamp,
25   *Algebraic Coding Theory*, McGraw-Hill, New York, 1968 and,
for example, S.Lin and D. J. Costello, *Error Control
Coding: Fundamentals and Applications*, Prentice-Hall,
Englewood Cliffs, N-J,1983, which are incorporated herein
by reference for all purposes.

30

       As is well understood within the art an (N,K) BCH or
RS code has K message symbols and N coded symbols, where
each symbol belongs to $GF(q)$ for a BCH code or $GF(q^m)$ for
an RS code. A binary (N,K) BCH code can correct up to t

errors with $N=2^m-1$, where N-K is greater than or equal to 2t. An (N,K) RS code can correct t errors and ρ erasures with $2t+\rho \leq N-K$. For binary BCH codes, an error can be corrected by finding the error locator and error

5 evaluator polynomials. In RS codes, an erasure is defined to be an error with a known error location. The steps performed by most RS decoder architectures for correction of errors can be summarised as follows:

(1) calculate the syndromes for the received code words,

10 (2) compute the error locator polynomial and the error evaluator polynomial,

(3) find the error locations, and

(4) compute the error values.

15 If both errors and erasures are corrected, the four steps are modified to be:

(1) calculate the syndromes and Forney syndromes from the received code words and the erasure locations,

(2) compute the errata locator polynomial and the errata

20 evaluator polynomial,

(3) find the errata locations, and

(4) compute the errata value.

In effect, the received data R(x) is provided to a

25 syndrome generator to generate a syndrome polynomial, S(x), which represents the error pattern of the code word from which errors can be corrected. The syndromes depend only on the error pattern, which is part of the syndrome, and not on the transmitted code word. A key equation

30 solver, which uses the well-known Berlekamp-Massey algorithm, is used to generate an error locator polynomial, σ(x), and an error evaluator polynomial, Ω(x). The error locator polynomial, σ(x), provides an indication of the locations of any errors. The error

35 evaluator polynomial provides the values of the located

2

errors. Typically, the error locator polynomial is processed by a Chien search engine to generate the roots, β_1^{-1}, of the polynomial, which roots provide the locations of any errors. The values of the errors are calculated

5 using the roots together with the error evaluator polynomial, $\Omega(x)$.

As is well understood by those skilled in the art there exists various architectures for solving the key

10 equation. Preferred architectures implement an inverse-free Berlekamp-Massey algorithm. Architectures for implementing the algorithm are typically implemented using a parallel processing architecture, which has the disadvantage of requiring the relatively large area and

15 having a relatively high logic circuit delay but with the advantage of a low processing latency. Alternatively, architectures might use a serial processing architecture, which has the disadvantage of a relatively large processing latency but the advantage of a relatively low

20 area and logic delays. However, in the context of a complete decoding system, the increased decoding latency requires increased data buffer sizes that, in turn, require additional space. It will be appreciated by those skilled in the art that the area of any given

25 processing architecture increases significantly as the number of finite field multipliers increases.

It is an object of embodiments of the present invention at least to mitigate some of the problems of

30 the prior art.

Summary of Invention

Accordingly, a first aspect of embodiments of the

35 present invention provides a data processing system

3

comprising a first arithmetic unit comprising at least one finite field multiplier and at least one finite field adder for selectively performing at least two finite field arithmetic calculations; the data processing system

5 comprising means to use a previous finite field arithmetic calculation result of the first arithmetic unit in a current finite field arithmetic calculation of the first arithmetic unit to determine respective coefficients of at least part of at least a first

10 polynomial.

Advantageously, the inverse-free Berlekamp-Massey algorithm according to embodiments of the present invention, when implemented, results in a relatively low

15 area and logic circuit delay as compared to a fully parallel implementation and a lower latency as compared to a fully serial implementation. It will be appreciated that this provides a very good compromise between area and processing latency. In particular, embodiments of

20 the present invention have a very favourable area-latency product as compared with practical implementations of prior art solutions.

Still further, embodiments of the present invention

25 use a relatively low number of finite field multipliers.

Embodiments provide a data processing system in which a first arithmetic operation of the at least two arithmetic operations comprises a first finite field

30 multiplication operation.

Preferably, embodiments provide a data processing system in which the first finite field multiplication

4

operation comprises calculating at least a first multiplication of $\delta\sigma^{(i-1)}(x)$ in a first clock cycle.

Embodiments provide a data processing system in which the first finite field arithmetic operation

5 comprises calculating at least a second multiplication operation of $\Delta^{(i)}x\lambda^{(i-1)}(x)$ in a second clock cycle.

Preferably embodiments provide a data processing system in which a second arithmetic operation of the at

10 least two arithmetic operations comprises a finite field addition operation. Preferably, the finite arithmetic addition operation comprises calculating at least part of $\delta\sigma^{(i-1)}(x)+\Delta^{(i)}x\lambda^{(i-1)}(x)$ as the current finite field arithmetic operation using $\delta\sigma^{(i-1)}(x)$ as at least part of

15 the previous finite field arithmetic operation

Embodiments provide a data processing system further comprising at least one further such arithmetic unit operable substantially in parallel with the first

20 arithmetic unit to calculate respective coefficients of at least part of at least a first polynomial. Preferably, the first polynomial comprises at least the coefficients of $\delta\sigma^{(i-1)}(x)+\Delta^{(i)}x\lambda^{(i-1)}(x)$.

25 Preferred embodiments provide a data processing system in which the at least two arithmetic calculations comprises a second finite field multiplication operation in a third clock cycle. The finite field second multiplication operation comprises calculating at least

30 one coefficient of a second polynomial.

Preferred embodiments provide a data processing system in which the second arithmetic operation comprises calculating at least $S_{i-j+1}\sigma_j^{(i)}$. Preferably, the second arithmetic operation comprises calculating at least part

5   of $S_{i+1}\sigma_0^{(i)}+S_i\sigma_1^{(i)}+...+S_{i-t+1}\sigma_t^{(i)}$.

Preferred embodiments provide a data processing system comprising at least $(t+1)$ arithmetic units operable substantially in parallel, each unit producing respective coefficients of at least one of a first

10 polynomial, $\sigma^{(i)}(x)=\delta\sigma^{(i-1)}(x)+\Delta^{(i)}x\lambda^{(i-1)}(x)$, and a second polynomial, $\Delta^{(i+1)}=S_{i+1}\sigma_0^{(i)}+S_i\sigma_1^{(i)}+...+S_{i-t+1}\sigma_t^{(i)}$.

Embodiments further provide a data processing system in which the first arithmetic unit is arranged to

15 calculate at least a respective part of at least part (coefficients) of a further polynomial. Preferably, the further polynomial is an error evaluator polynomial.

Preferred embodiments provide a data processing system in which the further polynomial comprises

20 calculating

$$\Omega(x) = S(x)\sigma(x) \bmod x^{2t}$$
$$= (S_0+S_1x+...+S_{2t-1}x^{2t-1}) \cdot (\sigma_0+\sigma_1x+...+\sigma_tx^t) \bmod x^{2t}$$
$$= \Omega_0+\Omega_1x+...+\Omega_{t-1}x^{t-1}, \text{ where}$$
$$\Omega_i = S_i\sigma_0+S_{i-1}\sigma_1+...+S_{i-t+1}\sigma_{t-1}, \text{ where } i=0,1,...,t-1.$$

25

Preferably, embodiments provide a data processing system in which the at least a respective part of at least part of the further polynomial comprises calculating:

30    $\Omega_i^{(j)}=S_i\sigma_0$, for $j=0$; and
      $\Omega_i^{(j)} = \Omega_i^{(j-1)}+S_{i-j}\sigma_j$, for $1 \leq j \leq i$.

6

Embodiments provide a Berlekamp-Massey algorithm processing unit comprising (t+1) finite field multipliers.

5   Embodiments also provide a Berlekamp-Massey processing element comprising (t+1) finite field processing units arranged, in a feedback arrangement, to perform at least (t+1) parallel operations; each parallel operation comprising at least two serial operations.

10

Embodiments provide a Berlekamp-Massey algorithm having an area-latency product of $7t^2+7t$.

Brief Description of the Drawings

15

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings in which:

figure 1 illustrates an embodiment of the present
20   invention for implementing an inverse-free Berlekamp-Massey algorithm;

figure 2 shows an embodiment of a multiply and accumulate cell;

figure 3 depicts a flowchart showing the
25   calculations performed using the multiple and accumulate cells of the first embodiment to calculate the error locator polynomial;

figure 4 illustrates the use of the architecture shown in figure 1 for calculating the error evaluator
30   polynomial; and

figure 5 shows a flowchart for calculating the error-evaluator polynomial.

Detailed Description of Preferred Embodiments

The theory underlying embodiments of the present invention will now be described. The key equation for
5 RS(N,K) and BCH(N,K) codes is:

$$S(x)\sigma(x) = \Omega(x) \bmod x^{2t}$$

where
10 $t = (N-K)/2$;
$S(x) = S_{2t-1}x^{2t-1} + S_{2t-2}x^{2t-2} + \ldots + S_0x^0$ is the Syndrome polynomial;
$\sigma(x) = \sigma_t x^t + \sigma_{t-1}x^{t-1} + \ldots + \sigma_0 x^0$ is the error-locator polynomial; and
15 $\Omega(x) = \Omega_{t-1}x^{t-1} + \Omega_{t-2}x^{t-2} + \ldots + \Omega_0 x^0$ is the error-evaluator polynomial.

The inverse-free Berlekamp-Massey Algorithm is a $2t$-step iterative algorithm:
20 $L^{(-1)} = 0$;
$\sigma^{(-1)}(x) = 1$;
$\lambda^{(-1)}(x) = 1$;
$\Delta^{(0)} = S_0$;
$\delta = 1$;

25

$$\textit{for } (i=0; \ i<2t; \ i=i+1)\{$$
$$\sigma^{(i)}(x) = \delta\sigma^{(i-1)}(x) + \Delta^{(i)}x\lambda^{(i-1)}(x);$$
$$\Delta^{(i+1)} = S_{i+1}\sigma_0^{(i)} + S_i\sigma_1^{(i)} + \ldots + S_{i-t+1}\sigma_i^{(i)};$$
$$\textit{if } (\Delta^{(i)} == 0 \ || \ 2L^{(i-1)} \geq i+1)$$

30
$$\{ \quad L^{(i)} = L^{(i-1)};$$
$$\lambda^{(i)}(x) = x\lambda^{(i-1)}(x);$$

$$\}$$

8

```
else
    {    L^(i) = i+1-L^(i-1);
```

$$\lambda^{(i)}(x) = \sigma^{(i-1)}(x);$$

$$\delta = \Delta^{(i)};$$

```
              }

}
```

where:

$\sigma^{(i)}(x)$ is the error-locator polynomial at step i;

$\lambda^{(i)}(x) = \lambda_t^{(i)}x^t + \lambda_{t-1}^{(i)}x^{t-1} + \ldots + \lambda_0^{(i)}x^0$ is the auxiliary polynomial at step i;

$\Delta^{(i)}$ is the step discrepancy at step i;

$L^{(i)}$ is the auxiliary degree variable at step i; and

δ is the previous step discrepancy.

Computation of the coefficients of the error-evaluator polynomial, $\Omega(x)$, follows directly after calculating the error-location polynomial, $\sigma(x)$, using

$$\Omega_i = S_i\sigma_0 + S_{i-1}\sigma_1 + \ldots + S_{i-t+1}\sigma_{t-1} \qquad i = 0, 1, \ldots, t-1$$

Figure 1 shows an architecture 100 for implementing an inversion-free Berlekamp-Massey algorithm according to an embodiment of the present invention. The architecture 100 comprises (t+1) multiply and accumulate cells together with associated muliplexers, registers and finite field adders. For the purpose of clarity, only four multiply and accumulate cells MAC_0 to MAC_t are shown.

Referring to the jth multiply and accumulate cell MAC_j, it can be appreciated that the cell has three inputs; namely, A, B and C, and a single output D for receiving values a_j, b_j, c_j, and producing an output value d_j respectively. The first value, a_j, is derived from a multiplexer 102 that selects one of a number of inputs

9

according to a control signal A'. In preferred embodiments, the inputs are the values $\Delta^{(i)}$, δ and S_{i-j+1}. The second input, B, derives its input value, b_j, from an associated multiplexer 104. The multiplexer 104 is

5   controlled to select one of a number of inputs according to a control signal B'. In a preferred embodiment, the second multiplexer 104 has three inputs bearing the signals or values $\lambda_{j-1}^{(i-1)}$, $\sigma_j^{(i-1)}$ and d_j. The third input, C, to the jth multiply and accumulate cell MAC_j preferably

10 derives its input value or input signal, c_j, from a third multiplexer 106 having a number of inputs that are selected according to an associated control signal C'. In a preferred embodiment, the input values or signals to the third multiplexer 106 comprise the output from the

15 multiply and accumulate cell MAC_j, that is, the signal or value d_j and the value 0.

The multiply accumulate cells MAC_0 to MAC_t are used to calculate the error-locator polynomial, $\sigma(x)$, in

20 effect, they are used to calculate

$$\sigma^{(i)}(x) = \delta\sigma^{(i-1)}(x) + \Delta^{(i)}x\,\lambda^{(i-1)}(x),$$

which can be decomposed to

$$\sigma_j^{(i)} = \begin{cases} \delta\sigma_0^{(i-1)}, & \text{for } j=0 \\ \delta\sigma_j^{(i-1)} + \Delta^{(i)}\lambda_{j-1}^{(i-1)} & \text{for } 1 \le j \le t \end{cases}$$

$$\Delta_j^{(i+1)} = \begin{cases} S_{i+1}\sigma_0 & \text{for } j=0 \\ \Delta_{j-1}^{(i+1)} + S_{i-j+1}\sigma_j^{(i)} & \text{for } 1 \le j \le t \end{cases}$$

25 It can be appreciated that the first multiply accumulate cell MAC_0 calculates the co-efficient, $\sigma_0^{(i)}$, for the ith iteration of $\sigma^{(i)}(x)$ while the remaining multiply accumulate cells MAC_1 to MAC_t calculate the co-

10

efficients $\sigma_1^{(i)}$ to $\sigma_t^{(i)}$ for the remaining 1 to t co-
efficients of $\sigma^{(i)}(x)$. The four numerals, 1 to 4, in the
circles, together with the adjacent expressions,
illustrate the progressive calculation of the various

5   parts of above expressions at corresponding cycles. The
first two encircled numerals relate to the calculation

of $\sigma_j^{(i)}$. The third and fourth numerals illustrate the

progressive calculation or accumulation of $\Delta_j^{(i+1)}$.

10   The control signals A', B' and C' are arranged in a
first cycle to calculate the first term of the expression
for $\sigma^{(i)}(x)$. Using the jth multiply and accumulate cell
MAC_j as an example, during a first cycle, the first
multiplexer 102 is controlled, in response to an

15   appropriate control signal, A', to select the input 108
bearing the signal or data value δ. The second
multiplexer 104 is controlled, in response to a
respective control signal, B', to select the second input
109 which bears the signal or value $\sigma_j^{(i-1)}$. The third

20   mulitplexer 106 is controlled, in response to an
appropriate control signal, C', to select its second
input, which bears the value 0.

       Figure 2 shows the internal structure of an

25   embodiment 200 a multiply and accumulate cell. Each of
the multiply and accumulate cells MAC_0 to MAC_t have the
structure shown in figure 2. It can be appreciated that
the embodiment 200 shown in figure 2 comprises a finite
field multiplier 202 arranged to multiply, using pre-

30   determinable modulo arithmetic, the first, A, and second,
B, inputs to the multiply accumulate cell 200. In a
preferred embodiment, the finite field multiplier
performs arithmetic over $GF(q^m)$, and, preferably, over
$GF(2^m)$. The output 204 from the finite field multiplier

11

202 is fed to a finite field adder 206 where it is combined with the value presented at the third input, C, of the multiply and accumulate cell 200. The finite field adder 206 operates over GF (q^m). In preferred embodiments, GF(q^m) = GF(2^m). A register 208 is provided to store the result of the finite field addition operation.

Referring back to figure 1 and, in particular, to the jth multiply and accumulate cell MAC$_j$, it can be appreciated, given the control signals A', B' and C' described above, that the jth multiply and accumulate cell MAC$_j$ calculates, in a first cycle, or in the ith cycle, the value $\delta\sigma^{(i-1)}{}_j$. It can be appreciated that the register 208 is used to store the first part of the calculation of the jth co-efficient of $\sigma^{(i)}(x)$, that is, $\sigma^{(i)}{}_j = \delta\sigma_j{}^{(i-1)}$.

During the next, that is, second, cycle, the calculation of $\Delta^{(i)}(x)\lambda^{(i-1)}(x)$ is performed. More particularly, the second part of the calculation of the jth co-efficient of $\sigma^{(i)}(x)$ is performed, that is, $\Delta^{(i)} \lambda^{(i-1)}_{j-1}$ is calculated.

The finite field adder is then arranged to add this result, that is, $\Delta^{(i)} \lambda^{(i-1)}_{j-1}$ to d$_j$, which is $\delta\sigma_j{}^{(i-1)}$, to give

$$\sigma_j{}^{(i)} = \delta\sigma_j^{(i-1)} + \Delta^{(i)} \lambda^{(i-1)}_{j-1}.$$

It will be appreciated that at the end of the second cycle, the value at node d$_j$ forms value σ_j, that is, $\sigma_j^{(i)}$, for the next cycle.

During the third cycle, the jth multiply and accumulate cell MAC_j is arranged to calculate the jth term, that is, $S_{i-j+1}\sigma_j^{(i)}$. It will be appreciated that the summation of the outputs d_j of each of the multiply accumulate cells MAC_0 to MAC_t represents the expression

$$\Delta^{(i+1)} = S_{i+1}\sigma_0^{(i)} + S_i\sigma_1^{(i)} + \ldots + S_{i-t+1}\sigma_t^{(i)}.$$

It will be appreciated that since each of the $(t+1)$ terms of this expression are computed in parallel, the calculation takes one clock cycle. Therefore, the embodiments require 3x2t clock cycles to calculate the error - locator polynomial, $\sigma(x) = \sigma_t x^t + \sigma_{t-1} x^{t-1} + \ldots + \sigma_0 x^0$.

Table One below shows the error-locator polynomial computation data flow for cycles 1, 2 and 3 of steps i and (i+1).

step	i			i+1		
cycle	1	2	3	1	2	3
a_j	δ	$\Delta^{(i)}$	S_{i-j+1}	δ	$\Delta^{(i+1)}$	S_{i-j+2}
b_j	$\sigma_j^{(i-1)}$	$\lambda_{j-1}^{(i-1)}$	d_j	$\sigma_j^{(i)}$	$\lambda_{j-1}^{(i)}$	
c_j	0	d_j	0	0	d_j	0
d_j	$S_{i-j}\sigma_j^{(i-1)}$	$\delta\sigma_j^{(i-1)}$	$\delta\sigma_j^{(i-1)} + \Delta^{(i)}\lambda_{j-1}^{(i-1)} \equiv \sigma_j^{(i)}$	$S_{i-j+1}\sigma_j^{(i)}$	$\delta\sigma_j^{(i)}$	$\delta\sigma_j^{(i)} + \Delta^{(i+1)}\lambda_{j-1}^{(i)} \equiv \sigma_j^{(i+1)}$
e	$\Delta^{(i)}$	...	...	$\Delta^{(i+1)}$	...	...
Δ	...	$\Delta^{(i)}$	$\Delta^{(i)}$	$\Delta^{(i)}$	$\Delta^{(i+1)}$	$\Delta^{(i+1)}$
σ_j	$\sigma_j^{(i-1)}$	$\sigma_j^{(i-1)}$	$\sigma_j^{(i-1)}$	$\sigma_j^{(i)}$	$\sigma_j^{(i)}$	$\sigma_j^{(i)}$
λ_{j-1}	$\lambda_{j-1}^{(i-1)}$	$\lambda_{j-1}^{(i-1)}$	$\lambda_{j-1}^{(i-1)}$	$\lambda_{j-1}^{(i)}$	$\lambda_{j-1}^{(i)}$	$\lambda_{j-1}^{(i)}$

Table 1: Error-Locator Polynomial Computation Data Flow

Referring to figure 3, there is shown a flowchart 300 for calculating the error-locator polynomial

13

according to an embodiment. At step 302, the variables used by the inverse-free Berlekamp-Massey algorithm are initialised. The first term, $\delta\sigma^{(i-1)}(x)$ of the expression

5
$$\sigma^{(i)}(x) = \delta\sigma^{(i-1)}(x) + \Delta^{(i)} x\lambda^{(i-1)}(x)$$

is calculated using each of the multiply and accumulate cells MAC_0 to MAC_t at step 304. The overall expression

10
$$\sigma^{(i)}(x) = \delta\sigma^{(i-1)}(x) + \Delta^{(i)} x\lambda^{(i-1)}(x)$$

is calculated at step 306. The multiply accumulate cells MAC_0 to MAC_t are used at step 308 to calculate each of the terms of $\Delta^{(i+1)} = S_{i+1}\sigma_0^{(i)} + S_i\sigma_1^{(i)} + ... + S_{i-t+1}\sigma_t^{(i)}$. A test is

15 performed, at step 310, to determine if $\Delta^{(i)} = 0$ or $2L^{(i-1)}$ is greater than or equal to i+1. If the determination is positive, effect is given to $L^{(i)} = L^{(i-1)}$ and $\lambda^{(i)}(x) = x\lambda^{(i-1)}(x)$ at step 312. If the determination at step 310 is negative, effect is given to $L^{(i)} = i+1-L^{(i-1)}$; $\lambda^{(i)}(x) = \sigma^{(i-1)}(x)$; and $\delta = \Delta^{(i)}$ at step 314. It will be appreciated that

20 effect is given to steps 312 and 314 using the control signals, D', to corresponding multiplexers 126 to 132, to select the appropriate values at the inputs to those multiplexers 126 to 132 to implement the equations shown in steps 312 and 314. The control variable, i, is

25 incremented by one at step 316. A determination is made at step 318 as to whether or not i is less than 2t, where $t = (N-K)/2$ for Reed-Solomon (N,K) or BCH(N,K) codes. If the determination at step 318 is positive, processing

30 resumes at step 304. However, if the determination at step 318 is negative, the calculations for the coefficients of the error-locator polynomial, $\sigma(x)$, are completed.

35   In a preferred embodiment, the computation of the

error-evaluator polynomial, $\Omega(x)$, can be calculated directly after the computation of the error-locator polynomial, $\sigma(x)$. It will be appreciated that t cycles are required to calculate the co-efficients of $\Omega(x)$. It
5 will be appreciated that having obtained $\sigma(x)$, from the key equation, which is

$$S(x)\sigma(x)=\Omega(x) \mod x^{2t},$$

10 one can derive

$$\Omega(x)=S(x)\sigma(x) \mod x^{2t}$$
$$=(S_0+S_1x+\ldots+S_{2t-1}x^{2t-1}).(\sigma_0+\sigma_1x+\ldots+\sigma_tx^t) \mod x^{2t}$$
$$=\Omega_0+\Omega_1x+\ldots+\Omega_{t-1}x^{t-1}.$$

15

It will, therefore, be appreciated that

$$\Omega_i=S_i\sigma_0+S_{i-1}\sigma_1+\ldots+S_{i-t+1}\sigma_{t-1}, \text{ where } i=0,1,\ldots,t-1.$$

20 The calculation of Ω_i is similar to that of $\Delta^{(i)}$. It can be appreciated that the same arrangement shown in figure 1 can be used to compute $\Omega(x)$ after that arrangement has been used to calculate $\sigma(x)$. It can be appreciated that Ω_i can be decomposed as follows

25

$$\Omega_i^{(j)}=S_i\sigma_0, \text{ for } j=0; \text{ and }$$
$$\Omega_i^{(j)}=\Omega_i^{(j-1)}+S_{i-j}\sigma_j, \text{ for } 1\leq j\leq i.$$

Therefore, referring to figure 4, the control
30 signal, A', for the first multiplexer 110 is arranged to select the third input, that is, S_i, as the input signal, a_0, to the lowest order multiply accumulate cell MAC_0. The second multiplexer 112 is arranged to select the second input, which bears the signal or data value for σ_0.

15

Therefore, the second input to the lowest order multiply accumulate cell MAC_0 is σ_0. The control signal, C', for the third multiplexer 114 is arranged to be 0. It will be appreciated that the output signal, d_0, will be given

5 by $S_i\sigma_0$, that is, $\Omega_i^{(0)}$. The second multiply and accumulate cell, MAC_1, calculates $S_{i-1}\sigma_1$. Therefore, the output of the finite field adder 120 is $\Omega_i^{(1)} = \Omega_i^{(0)} + S_{i-1}\sigma_1$. Similarly, the jth multiply and accumulate cell, MAC_j, is arranged to select the third input 116 of the first multiplexer 102

10 and the second input 109 of the second multiplexer 104 to produce, at the output, d_j, the signal or data value $d_j = S_{i-j}\sigma_j$. Hence, the output of the finite field adder 122 is $\Omega_i^{(j)} = \Omega_i^{(j-1)} + S_{i-j}\sigma_j$. The output values of each of the multiply and accumulate cells are shown adjacent to the

15 encircled 1. The encircled "2"s illustrate the progressive accumulation, by the finite field adders 120 to 124, of the d_0, d_1,...,d_j,... and d_t values to produce the overall expression for

20 $\Omega_i = S_i\sigma_0 + S_{i-1}\sigma_1 + ... + S_{i-j}\sigma_j + ... + S_{i-t+1}\sigma_{t-1}$.

It will be appreciated that the calculation of Ω_i takes a single clock cycle. Therefore, the overall calculation of $\Omega(x)$ will require t clock cycles since the

25 co-efficients, Ω_i, $i=0,...,t-1$, are calculated in series using the arrangement 400 shown in figure 4 with each term, $S_{i-j}\sigma_j$, for $0 \le j \le i$, being calculated in parallel using sufficient of the multiply and accumulate cells MAC_0 to MAC_t.

30

It can be appreciated that the error-evaluator polynomial computation data flow is as shown in table two below.

16

step	i	$i+1$
a_j	S_{i-j}	S_{i-j+1}
b_j	σ_j	σ_j
c_j	0	0
d_j	$S_{i-j-1}\sigma_j$	$S_{i-j}\sigma_j$
e	Ω_{i-1}	Ω_i
σ_j	σ_j	σ_j

Table 2: Error-Evaluator Polynomial Computation Data Flow

Therefore, preferred embodiments for implementing the inverse-free Berlekamp-Massey algorithm comprise
5   $(t+1)$ finite field multipliers and require a total of $7t$ clock cycles to compute the error-locator polynomial and the error-evaluator polynomial. The architecture 100 shown in figure 1 can be used for detecting and correcting errors. It can also be applied to the
10   correction of errors and erasures. Since the area of the architecture 100 shown in figure 1 is proportional to the number of finite field multipliers, that area is proportional to $(t+1)$. Furthermore, the latency of the architecture 100 shown in figure 1 is given by $7t$.
15   Therefore, the area-latency product of embodiments is given by $7t^2+7t$, which compares favourably with many fully parallel or fully serial implementations of the inverse-free Berlekamp-Massey algorithm. Still further, it will be appreciated that the architecture 100 shown in figure
20   1 does not require a relatively complex controller.

Referring to figure 5, there is shown a flowchart
500 for calculating the error-evaluator polynomial having obtained the error-locator polynomial. The control
25   variable, i, is set to zero at step 502. The expression

17

$$\Omega_i^{(j)} = \begin{cases} S_i \sigma_0, & \text{for } j=0 \\ \Omega_i^{(j-1)} + S_{i-j} \sigma_j & \text{for } 1 \le j \le \end{cases}$$

is performed at step 504 using as many of the (t+1) multiply accumulate cells MAC_0 to MAC_t as is necessary. The control variable, i, is incremented by one at step

5   506. A determination is made, at step 508, as to whether or not i is less than or equal to t-1. If the determination is positive, processing continues at step 504 where the next co-efficient Ω_i of $\Omega(x)$ is calculated. If the determination at step 508 is negative, the

10   calculation of the error-evaluation polynomial is deemed to be complete.

It will be appreciated that the above embodiment can be readily adapted to take into account erasures. For

15   example, if both errors and erasures are to be corrected, the steps are modified to be:

   (1) calculate the syndromes and Forney syndromes from the received code words and the erasure locations,

   (2) compute the errata locator polynomial and the errata

20   evaluator polynomial,

   (3) find the errata locations, and

   (4) compute the errata value.

It will be appreciated by those skilled in the art

25   that values of S for undefined subscripts, that is, subscripts that are less than zero, are preferably arranged to be processed as zeros by embodiments of the present invention. It will also be appreciated that this applies equally to calculations relating to both the

30   error locator polynomial and the error evaluator polynomial.

The reader's attention is directed to all papers and documents which are filed concurrently with or previous to this specification in connection with this application and which are open to public inspection with this specification, and the contents of all such papers and documents are incorporated herein by reference.

All of the features disclosed in this specification (including any accompanying claims, abstract and drawings) and/or all of the steps of any method or process so disclosed may be combined in any combination, except combinations where at least some of such features and/or steps are mutually exclusive.

Each feature disclosed in this specification (including any accompanying claims, abstract and drawings) may be replaced by alternative features serving the same, equivalent or similar purpose, unless expressly stated otherwise. Thus, unless expressly stated otherwise, each feature disclosed is one example only of a generic series of equivalent or similar features.

The invention is not restricted to the details of any foregoing embodiments. The invention extends to any novel one, or any novel combination, of the features disclosed in this specification (including any accompanying claims, abstract and drawings), or to any novel one, or any novel combination, of the steps of any method or process so disclosed.